Performance Evaluation of a Fuzzy Logic Based Kalman Filter for Target Tracking

S. K. Kashyap[+] and J. R. Raol[++]

[+] Scientist, MSDF Lab, Flight Mechanics and Control Division
[++]Scientist and Head, Flight Mechanics and Control Division
National Aerospace Laboratories- Bangalore-560017
Email: sudesh@css.nal.res.in; jrraol@css.nal.res.in

ABSTRACT

The Kalman filter is globally accepted by estimation community and frequently applied in many real: time applications such as tracking, navigation, guidance, control process etc. The optimality of Kalman filter depends upon how accui-ate the mathematical models for actual dynamical system and measurement device are known. The performance of this filter also depends on tuning parameters such as process noise variance (Q), and measurement noise variance (R). However, there can be situations when exact mathematical models may not he known or too difficult to model. In such cases, it is widely experienced that modelling en-ors are often coinpensated by overloading the tuning parameters of the filter by using trial and error method. But this becomes extra burden to filter designer and time-consuming task. To tackle such problems, Fuzzy logic can be one of the promising solutions that uses an intuitive experience based-approach for problems that are too difficult to model mathematically and for filters difficult to tune properly. This paper considers the combination of Fuzzy logic and Kalman filter that have traditionally been considered to be radically different. The former is considered heuristic and the latter statistical filtering. Two schemes such as Kalman filter and Fuzzy Kalman filter are applied for target tracking application and their performances evaluated using several numerical examples. The approach is relatively novel. Also comparison with one of the existing adaptive tuning algorithms is carried out. The performance is evaluated using certain error-based criteria.

Key Words: Kalman filter, Fuzzy logic, Fuzzy Kalman filter, Target tracking

1. INTRODUCTION

In order to estimate the unknown state o f a given dynamic system using noisy measurements (that contain directly or indirectly some information about a system) originating from sensor, state estimator is employed to produce an accurate estimate as close as possible to true system state. It is widely experienced that Kalman filter (KF) is a most promising algorithm for recursive estimation of any linear system being observed by single or group of sensors (with linear model). However, the accuracy of KF is based on how accurate the mathematical models for actual dynamic system and measurement device are known. There can be situations when exact mathematical model may not be exactly known or too difficult to model. The model mismatch is often compensated by a proper selection (through trial & error) of tuning parameters e.g. pi-ocess noise covariance Q. This approach provides sub-optimal solution to a given problem.

To tackle such problems, fuzzy logic can be used as one of the promising solutions that uses an intuitive experience based-approach for problems that are too difficult to model mathematically and for filters difficult to tune properly. The aim of fuzzy logic is to combine scientific rigour with expert intuition. This combination allows a more rigorous capture of a priori information through fuzzy logic when formulating the optimal stochastic estimation method of Kalman filtering. For example, it is easy to predict rain in monsoon season. But it is difficult to say which day it will occur. That can be forecast with a prior available statistics and information pertaining to humidity, temperature, etc. However the correctness of rain prediction depends upon how accurately we model the monsoon. Application of fuzzy logic in rain prediction assumes that monsoon is so difficult to model that its forecasting requires both intuitive experience and statistics.

The other application of fuzzy logic, developed by Hitachi [1], is for automatic operation of subway trains. This fuzzy system adopted train drivers' experiences in controlling the velocity, acceleration and braking systems. The rules of logic were derived from interviews conducted with train drivers. The rules were obtained in view of improving the safety, the convenience, the energy consumption, the travel time, and the precision of stoppage at the subway platform. Their implementation resulted in fewer number of applications of brakes, which led to lower energy consumption and improved ride quality for passengers. Like that there are several examples that use combination of fuzzy logic and KF.

This paper considers the combination of fuzzy logic and KF that have traditionally been considered to be radically different. The former is considered heuristic and the latter as statistical filtering. Two schemes such as KF and Fuzzy Kalman filter (FKF) are applied for target tracking application and their performances evaluated.

2. CONVENTIONAL KALMAN FILTER

Conventional KF is an optimal, recursive, and minimum variance filter and is often used in real-time target tracking applications. The time varying gain of KF decides how much weightage should be given to present observation i.e. if data is highly contaminated with noise then automatically less weightage is given to that data and filter depends on the model of target (i.e. state propagation). However. if data is less noisy then more weightage is assigned to that data and estimated state is combination of state predicted (through target model) and observation data. The equations needed to implement KF are as follows:

State and Covariance Propagation

$$\tilde{X}(k+1/k) = F\hat{X}(k/k) \tag{1}$$

$$\tilde{P}(k+1/k) = F\hat{P}(k/k)F^T + GQG^T \tag{2}$$

State and Covariance Update

$$\left.\begin{aligned} K &= \tilde{P}(k+1/k)H^T S^{-1} \\ S &= H\tilde{P}(k+1/k)H^T + \boldsymbol{R} \end{aligned}\right\} \tag{3}$$

$$\left.\begin{aligned} \hat{X}(k+1/k+1) &= \tilde{X}(k+1/k) + Ke(k+1) \\ e(k+1) &= Z(k+1) - H\tilde{X}(k+1/k) \end{aligned}\right\} \tag{4}$$

$$\hat{P}(k+1/k+1) = [I - KH]\tilde{P}(k+1/k) \quad (5)$$

where, F is state transition matrix, G is process noise gain matrix, Q is process noise covariance matrix, $\tilde{X}$ is predicted state vector, $\tilde{P}$ is predicted state covariance matrix, H is measurement model, R is measurement noise covariance matrix, K is Kalman gain, S is innovation covariance matrix, e is innovation sequence vector, $\hat{X}$ is estimated state vector, and P is estimated state covariance matrix.

3. FUZZY LOGIC AND FUZZY KALMAN FILTER

Zadeh's Fuzzy Logic (FL) concept **[2]** facilitates modeling the conditions that are inherently imprecisely defined. Fuzzy teclmiques in the fonn of approximate reasoning provide decision support and expert system with powerful reasoning capabilities. In the past few decades, FL techniques have been used in varieties of applications such as i) image-analysis (e.g. detection of edges, feature extraction, classification, and clustering), ii) parameter estimation of unknown dynamic systems, e.g. aircraft, iii) home appliances, e.g. washing machine, air conditioning systems, and iv) decision fusion, e.g. situation assessment. 'Fuzzy logic has inherent abilities to mimic the human mind so that it can be deployed for reasoning that are approximate rather than exact.

In this paper, we use fuzzy logic concept at measurement update level [3]. This process is called fuzzy state conelator (FSC) and the filter as fuzzy Kalman filter (FKF). Except eq. (4), all the other equations, eqs. (1)-(3) and eq. (5), of FKF are the same as those of KF. The modified eq. (4) is given by

$$\hat{X}(k+1/k+1) = \tilde{X}(k+1/k) + KC(k+1) \quad (6)$$

where. $C(k+1)$ is tlie correction vector as an output from FSC and is a nonlinear function of the innovation vector e. In present case, we use FKF in target tracking application and assume that only position (x-y axes) measurements of target are available. Thus, the correction vector consists of modified innovation sequence for x and y axes i.e.

say, $C(k+1) = \left[c_x(k+1) \quad c_y(k+1)\right]$.

To find $C(k+1)$, the innovation vector e is first separated into its x and y components, e_x and e_y. We assume that target motion in each axis is independent. and develop the FSC for tlie x direction and then generalize the result to include y direction. The FSC consists of two inputs (i.e. e_x and $\dot{e}_x$) and single output $c_x(k+1)$, where $\dot{e}_x$ is computed by

$$\dot{e}_x = \frac{e_x(k+1) - e_x(k)}{T} \quad (7)$$

where, T is the sampling period in seconds

Realization of Fuzzy State Correlator through FIS [2]

FSC is a Fuzzy Inference System (FIS). In order to understand FIS, let us define a singleton Fuzzy rule as "IF u is A, THEN v is B". The IF part of the rule, "IF u is A," is called the *antecedent* or *premise*, while THEN part of the rule, "v is B", is called the *consequent* or conclusion. For any fuzzy logic based mechanism or system, the core part is inference engine. The inference engine (through fuzzy implication operation) defines mapping from input fuzzy sets into output fuzzy sets. It determines the degree to which the *antecedent* is satisfied for each rule. If the *antecedent* of a given rule has more than one clause (e.g. "IF u_1 is A_1 AND u_2 is A_2, THEN v is B"), fuzzy operators (t-norm/s-norm) [2] are applied to obtain one number that represents the result of the *antecedent* for that rule. Inference engine can take different forms depending on the manner in which inference (rule) is defined.

It is also possible that one or more rules may fire at the same time. In such a case outputs of all rules are *aggregated* i.e. fuzzy sets that represent the output of each rule are combined into single fuzzy set. One important aspect of FIS is that fuzzy rules are fired in parallel and the order in which rules fired does not affect the output. Fig. I shows a schematic of a FIS for multi-input, single-output system. It can be seen from the figure that other components of FIS are fuzzifier, rule base, and defuzzifier. The fuzzifier maps input numbers into corresponding memberships. This is very much essential to activate rules that are in terms of linguistic variables. The fuzzifier takes input values and detemiines the degree to which they belong to each of fuzzy sets via membership functions. The rule base contains linguistic rules that are provided by experts. The defuzzifier maps output Fuzzy sets into crisp number. The most popular defuzzification method is the centroid.

Steps of Fuzzy Inference Process

Consider the i^{th} fuzzy rule (with more than one part in antecedent) for MISO (Multi Inputs Single Output) system defined as

$$R^i: \text{IF } u \text{ is } T_u^i \text{ AND } v \text{ is } T_v^i \text{ THEN } w \text{ is } T_w^i \tag{8}$$

where u, v, and w are the fuzzy or linguistic variables whereas T_u, T_v, and T_w are their linguistic values (e.g. LOW, HIGH, LARGE etc.). In order to get the crisp output using FIS. following steps are needed.

Step 1 : fuzzify the inputs u, and v using membership functions ($\mu^i(u)$, and $\mu^i(v)$) for i^{th} rule.

Step 2 : Since antecedent part of every rule has more than one clause, Fuzzy logic operator is used to resolve the antecedent to a single number between 0 and I that gives degree of support (or firing strength) for i^{th} rule . The firing strength can be expressed by

$$a^i = \mu^i(u) * \mu^i(v) \tag{9}$$

where, * represents triangular norm [2]. The most popular t-norms used are:

$a' = min(\mu^i(u), \mu^i(v))$ - standard intersection (10a)

$a' = \mu^i(u) \bullet \mu^i(v)$ - algebraic product (10b)

Step 3 Apply implication method to shape the consequent part (the output Fuzzy set) based on the antecedent. The input to the implication process is a single number (a) given by the antecedent and the output is a fuzzy set. The most coninionly used methods are : i) min-operation rule of fuzzy implication (Manidani) and ii) product operation rule of fuzzy implication represented by equations (11) and (12) respectively.

$$\mu^i(w)' = min(\alpha^i, \mu^i(w)) \quad (11)$$

or

$$\mu^i(w)' = \alpha^i \bullet \mu^i(w) \quad (12)$$

Step 4 : Since more than one rule (i.e. more than one output fuzzy set) can he fired at a time, we need to combine the corresponding output fuzzy sets into single composite fuzzy set. This process is known as *aggregation*. The inputs to aggregation process are outputs of implication process and output of aggregation process is a single fuzzy set that represents the output variable. The order in which rules are fired does not bother aggregation process. The most commonly used aggregation method is the *max* method.
Suppose rule 3 and rule 4 are fired at a time, **then** the composite output fuzzy set could he expressed as

$$\mu(w) = max(\mu 3(w)', \mu^4(w)') \quad (13)$$

It should be kept in mind that eq. (13) represents the final output membership curve or function.

Step 5 : In order to get crisp value of output variable w, defuzzification process is used. The input to this process is the output ofaggregation process i.e. eq. (13) and output is a single crisp number. Several methods for defuzzification process are described in [2].

Design of a Fuzzy State Correlator

The *antecedent* membership functions that define the Fuzzy values for inputs e_x and $\dot{e}_x$, are shown in figures 2-3 respectively. Similarly membership functions for output c_x are shown in figure 4. The labels used in Linguistic variables to define membership functions are LN (large negative), MN (medium negative), SN (small negative), ZE (zero en-or), SP (small positive), MP (medium positive), and LP (large positive). The rules for the inference in FIS are created based on the past experiences and intuitions. For example, one such rule is:

IF e_x is LP **AND** $\dot{e}_x$ is LP THEN c_x is LP (14)

This rule is created based on the fact that having e_x and $\dot{e}_x$ **with** large positive values indicate an increase in innovation sequence at faster rate. The future value of e_x (and therefore $\dot{e}_x$)

can be reduced by increasing the present value of $c_x (\approx Z - H\tilde{X})$ with a large magnitude. Table 1 summarizes the 49 rules needed to implement FSC. Output c_x at any instant of time can be computed using the inputs e_x and $\dot{e}_x$, input membership functions, rules mentioned in Table 1, fuzzy inference engine, aggregator, and defuzzification. Figure 5 illustrates the three dimensional surface view of **FIS** for present application. The properties of Fuzzy inference system used in present work are:

- FIS type — mamdani
- AND operator — min
- OR operator — max
- Implication — min
- Aggregation — max
- Defuzzification — centroid

4. RESULTS AND DISCUSSIONS

Example 1

The target data in x-direction is generated using constant acceleration model with process noise increment. With sampling interval $T=0.1$ seconds, a total of $N=100$ scans are generated. The data simulation proceeds with the following assumed parameter values and equations

- initial states of target $(x, \dot{x}, \ddot{x})$ are $(0\,\text{m}, 100\,\text{m/s}, 0\,\text{m/s}^2)$ respectively
- process noise variance $Q=0.0001$
- constant acceleration model

$$F = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$$G = \begin{bmatrix} T^3/6 & T^2/2 & T \end{bmatrix} \tag{16}$$

- target state equation

$$X(k+1) = FX(k) + Gw(k) \tag{17}$$

where, k is the scan number and w is the process noise, white and Gaussian, with zero mean and covariance Q.

- measurement equation

$$Z_m(k) = HX(k) + v(k) \tag{18}$$

$$H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{19}$$

where. H is the observation matrix and v is the measurement noise, white and Gaussian, with zero mean and covariance $R = \sigma^2$ (σ is the standard deviation of noise with a value of 10meter).

The initial conditions, F, G, H, Q, and R for both the filters are kept same. The initial state vector $\hat{X}(0/0)$ is kept close to true initial states. If the initial state of filter is far from to truth then $\hat{P}(0/0)$ should be large enough so that filter can rely more on measurement data to correct the state estimate as fast as possible. On other hand, $\hat{P}(0/0)$ should be kept small if the initial state of filter is close to true, which indicates that filter can rely more on process model as compared to measurement model.

The results for both the filters are compared in terms of true and estimated states, and states errors with bounds at every scan number. Figure **6** shows the comparison of true and estimated positions, velocities and accelerations of target in x-direction. Every effort was made to tune the KF properly. It is clear from the plots that KF shows initial transient and takes more time to settle down as compared to FKF. Figure 7 illustrates the comparison of state errors with $\pm 2\sqrt{\hat{P}}$ bounds. The state errors are within bounds for both the filters. However, it observed from the plots that FKF performs much better compared to KF.

Consistency Check of KF and FKF

The normalized cost function (CF) is computed using following formula

$$CF = \frac{1}{N}\sum_{k=1}^{N} e(k)S(k)^{-1}e(k)^{T} \tag{20}$$

where, e is innovation sequence vector, S is innovation covariance matrix, and N is total number of scans generated (number of data points).

Consistency Criteria: The filter performance is consistent when its normalized CF, eq. (20) is equal to the dimension of measurement vector.

The consistency check of KF and FKF is done for the following cases

case 1 : only position as measurement (R = 100)
case 2 : only position and velocity as a measurements (R = diag(100,1))
case 3 : only position, velocity and acceleration as a measurenients (R = diag(100,1,0.01))

Using eq. (20). cost functions of KF and FKF for the above cases are computed as shown in Table 2. We see that the cost function for KF is close to the theoretical value, therefore filter performance is consistent. For FKF. its cost function is slightly away from the theoretical prediction but still comparable with KF. This means that FKF can also be treated as consistent filter.

The performance of both the filters in terms of states errors are compared for all the three cases. Figures 8-10 illustrate position, velocity. and acceleration errors. The following observations are made:

- state error reduces for both the filter as more number of observables are used
- FKF shows better performance than KF for all the three cases

Example 2

Both the filter schemes are also evaluated for the state estimation of target data in x and y directions. We assume that target motion in each **axis** is independent. With that assumption, target data in y direction is simulated (using same models/parameters as used in x-direction) with an initial states of $y, \dot{y}, \ddot{y}$ = [0 m, -100 m/s, -10 m/s^2]. Due to inclusion of y-direction data, matrices such as F, G, H, and R used is both the filter are as follows

$$F = \begin{bmatrix} 1 & T & T^2/2 & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & T^2/2 \end{bmatrix} \quad (21)$$

$$G = \begin{bmatrix} T^3/6 & T^2/2 & T & T^3/6 & T^2/2 & T \end{bmatrix} \quad (22)$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (23)$$

$$R = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \quad (24)$$

The same FSC is used for y-direction. For y-axis, we observe similar performance of both filters as reponed in figures 6-7 and not shown here.

The perfomiances of both schemes are also compared in terms of RSSPE (root sum square position error) $= \sqrt{(x(k/k)-\hat{x}(k/k))^2+(y(k/k)-\hat{y}(k/k))^2}$), RSSVE (root sum square velocity error) $= \sqrt{(\dot{x}(k/k)-\hat{\dot{x}}(k/k))^2+(\dot{y}(k/k)-\hat{\dot{y}}(k/k))^2}$ and KSSAE (root sum square acceleratioii error) $= \sqrt{(\ddot{x}(k/k)-\hat{\ddot{x}}(k/k))^2+(\ddot{y}(k/k)-\hat{\ddot{y}}(k/k))^2}$. Figures 11-13 compare the RSSPE, RSSVE, and RSSAE computed using true and estimated states for both the filters. Although the performance of the KF is satisfactory and acceptable, the FKF performs better than the KF.

Example **3**

For the same target data (i.e. x & y axes), we also compare the performance of KF with FKF for two cases: i) when all **49** rules are taken (see Table 1) into consideration, and ii) only 4 rules are used (see Table 3). The membership functions for new FSC are shown in figures 14-16. Figure 17 shows the 3D surface view of inputs-output mapping for new FSC. Figure 18 compares the RSSPE of FKF for these two cases. It is very much clear that filter with 49 rules shows better performance than the filter with **4** rules only, although the performance with 4 rules is acceptable. This indicates that in order to have a good FKF we need sufficient rilles to get continuous and smooth inputs-output mapping.

Example **4**

Up to this point, we have discussed the application of FKF for tracking non-maneuvering target. In order to use FKF for tracking maneuvering target, we need to re-design the FSC to capture the various possible maneuver modes of the target. Re-designing of FSC involves the following:

- proper selection of membership functions of inputs and output
- tuning of selected membership functions
- proper selection of Fuzzy operators (e.g. t-norm & s-norm)
- proper selection of Fuzzy implication, aggregation and defuzzification methods

In present paper, we use MATLABB based functions such as 'genfis1()' to create initial FSC and 'anfis()' to tune it. These functions require training and check data which we obtained from true and measured target positions. Figure 19 shows the procedure to get tuned FSC.

Training and Check data

The target states are simulated using a third order. kinematic model with p.ocess noise acceleration increments (eqs.21-22) and additional arbitrary accelerations.. Measurements are obtained using (eqs. **18-23).** With a sampling interval of 1 sec, a total of 150 scans are generated. The data simulation proceeds with tlie following assumed parameter values

- initial states $(x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y})$ of target are (100, 30, 0, 100, 20, 0) respectively
- process noise variance Q=0. 1. It is assumed that $Q_{xx} = Q_{yy} = Q$
- measurement noise variance R=25. It is assumed that $R_{xx} = R_{yy} = R$

It is further assumed that the target has an additional acceleration of (x_{acc}, y_{acc}) at scans 25 and 100 and an acceleration of $(-x_{acc}, -y_{acc})$ at scans *50* and *75*. Data simulation is carried out using the base state model defined in Eq. (17) with process noise vector w (which is a 2x1 vector) modified to include additional accelerations at the specified scan points, to induce a specific maneuver as follows:

$$\left.\begin{aligned} w(1) &= guass() * \sqrt{Q_{xx}} + x_{acc} \\ w(2) &= guass() * \sqrt{Q_{yy}} + y_{acc} \end{aligned}\right\} \text{ at scans 25 and 100} \qquad (25)$$

$$\left.\begin{aligned} w(1) &= guass() * \sqrt{Q_{xx}} - x_{acc} \\ w(2) &= guass() * \sqrt{Q_{yy}} - y_{acc} \end{aligned}\right\} \text{ at scans 50 and 75} \qquad (26)$$

At the other scan points, the vector w is simply defined by

$$\left.\begin{aligned} w(1) &= guass() * \sqrt{Q_{xx}} \\ w(2) &= guass() * \sqrt{Q_{yy}} \end{aligned}\right\} \quad (27)$$

Acceleration magnitudes of $x_{acc} = -9*9.8m/s^2$ and $y_{acc} = 9*9.8m/s^2$ are used in Eqs. ***(25)*** and ***(26).*** The function $guass()$ uses a central-limit theorem to generate Gaussian random numbers with mean 0 and variance 1.

First we create the initial FSC for x-axis and tune it using inputs u_x^1, u_x^2 and output o_x obtained using following equations:

$$u_x^1(k) = z_x(k) - x(k) \quad (28)$$

$$u_x^2(k) = \frac{u_x^1(k) - u_x^1(k-1)}{T} \quad (29)$$

$$o_x(k) = m * u_x^1(k) \quad (30)$$

where x , z_x are true and measured target x-position respectively, m is the unknown parameter and should be properly selected based on maneuver capability of a particular target of interest. We kept $m = 2$ for present case. We take half of the total simulated points for training and remaining half as a checking data set. The same procedure is followed to get tuned FSC for y-axis. The trained FSC are then plug-in to FKF and its performance is compared with the KF for following two cases:

case A: mild maneuver data

In order to generate mild maneuver data, example 4 is used with minor modification in arbitrary acceleration injection points. A total of 17 scans are generated. Accelerations are injected at scans 8 ($x_{acc} = 6m/s^2$ and $y_{acc} = -6m/s^2$) and 15 ($x_{acc} = -6m/s^2$ and $y_{acc} = 6m/s^2$) only.

case B: evnsive maneuver data

In order to generate evasive maneuver dara, example 4 is used with the same points for arbitrary acceleration injection but with a maneuver magnitude of $40*9.8m/s^2$ (i.e. instead of $9*9.8m/s^2$).

The results for these cases are obtained for 100 Monte-Carlo runs. The initial state vectors of KF and FKF are same and kept close to initial true states. Initial state error covariance matrices for both the filters are kept to unity. Figure 20 compares the measured, true and estimated x-y target positions for case **A.** The comparison of estimated trajectories from KF and FKF with true and measured trajectory is reasonably good. Some discrepancies are observed in the maneuvering phase of the flight where FKF exhibits better performance than KF. Similar observations are made for case B (results not shown). In Figure **21** of case B. the IISSPE, RSSVE, and RSSAE for KF are found to be large compared to those for FKF,

indicating that KF is unable to satisfactorily track the target during tlie maneuver phase compared to FKF.

Example 5

In this example, the perfomiance of FKF is compared with KF and adaptive Kalman filter (AKF) using simulated data of "Example 4/case **A:** mild maneuver". The equations of AKF are same as those of KF (mentioned in section *2)* but with varying process noise covariance Q, estimated online using Maybeck method [4]. The equations required to estimated Q are given by

$$Q(k) = G^{\#}\left[\hat{P}^{-} - F\hat{P}^{+}F^{T}\right](G^{\#})^{T} \tag{31}$$

$$\hat{P}^{-} = K(k)\hat{A}(k)*(H^{T})^{\#} \tag{32}$$

$$\hat{P}^{+} = \hat{P}^{-} - K(k)H\hat{P}^{-} \tag{33}$$

$$\hat{A}(k) = \frac{1}{WL}\sum_{j=k-WL+1}^{k} e(j)e(j)^{T} \quad ; \; k \geq WL \tag{34}$$

where. # stands for pseudoinverse, e is the innovation sequence vector computed using eq.(4), k is the scan number and WL (=5 for present case)is the window length. It is important to note that online value of Q can be made available to AKF only from WL^{th} scan which means tliat accuracy of Q will be depends upon its initial guess(for k=1 to WL^{th}-1 scans) chosen by a filter designer.

Figure *22* compares the measured, true and estimated x-y target positions. From the figure it is clear tliat AKF exhibits better tracking accuracy, especially during target maneuver, as compared to KF but still it has slightly degraded perfomiance as compared to FKF. For non-maneuvering phases of target motion. AKF and KF perform almost similar. During the maneuvering portion, it is found that the magnitude of online Q increases and so Kalnian gain, hence automatically more weight is assigned to measurement model which aids in convergence of the estimated states to true values at much faster rate than as seen for KF only.

Additionally, a sensitivity study of AKF w.r.t. different values of WL (5 and 10) is carried out and its performance in terms of RSSPE (see figure 23) is compared with KF and FKF. It is observed from the figure tliat RSSPE of AKF during maneuvering phase is high for WL=10 as compared to for WL=5. This could be due to non-availability of online Q and with an assumption that its initial guess is not sufficient at a time when actual maneuver starts i.e. at k=8. An appropriate selection of WL requires *a priori* knowledge about when the target will maneuver first, that may not be available for enemy target, puts a limitation on AKF as compared to FKF.

5. CONCLUSIONS

A Fuzzy logic based KF is used to generate the correction vector needed to update the state estimate. Both tlie filters are compared in terms of estimated states with true values, states error with bounds, RSSPE. RSSVE, and RSSAE. In order to track maneuvering target, FSC is re-designed using training and clieck data sets obtained from true and measured target positions. The trained FSC is use in FKF and its performance is compared with KF for two

cases (A & B) i.e. target with mild and evasive maneuver. From the results it is clear that FKF gives comparatively better perfomiance than KF. It is suggested that a sufficient number of rules, to develop an efficient FSC and in turn good FKF, should be used. Additionally, KF and FKF is compared with adaptive Kalman filter and found that although AKF shows better result as compared to KF but still not as good as FKF especially during maneuvering phase of target. It is also observed that AKF is quite sensitive to selection of window length required for online estimation of process noise covariance.

6. REFRENCES

1. Passino, K.M. and Yurkovich S., *Fuzzy Control*, Addison-Wesley, 1998.

2. S.K. Kashyap and J.R. Raol, "*Unification and interpretation of fuzzy set operations*", NAL PD FC 0502, FMCD, NAL, Bangalore, March 2005.

3. Klein **L.A.,** Sensor and Data Fusion, *A Tool for Information Assessment and Decision Making*, SPIE Press, Washington, **USA,** 2004.

4. Peter S. Maybeck, *Stochastic Models, Estimation, and Control*, Vol. 2, Academic Press, Inc. (London) Ltd., 1982.

Table 1: Fuzzy Associated Memory (FAM) with 49 rules – example I

$\dot{e}_x$ \ e_x	LN	MN	SN	ZE	SP	MP	LP
LN	LN	LN	MN	MN	MN	SN	ZE
MN	LN	MN	MN	MN	SN	ZE	SP
SN	MN	MN	MN	SN	ZE	SP	MP
ZE	MN	MN	SN	ZE	SP	MP	MP
SP	MN	SN	ZE	SP	MP	MP	MP
MP	SN	ZE	SP	MP	MP	MP	LP
LP	ZE	SP	MP	MP	MP	LP	LP

Table 2: Comparison of cost function for KF and FKF – example I

	Normalized cost function		
	true	computed	
		KF	FKF
case 1	1	0.93	0.85
case 2	2	1.96	1.84
case 3	3	2.94	2.85

Table 3: **Fuzzy** Associated Memory (FAM) with 4 rules – example 3

$\dot{e}_x$ \ e_x	LN	LP
LN	LN	ZE
I, P	ZE	LP

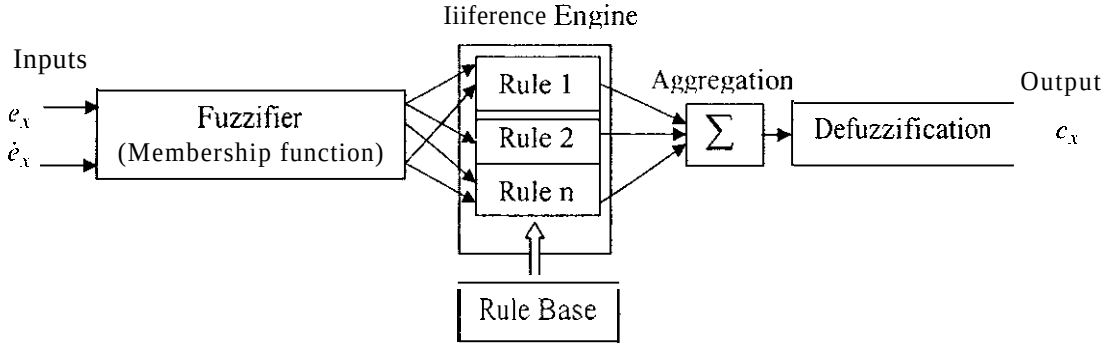
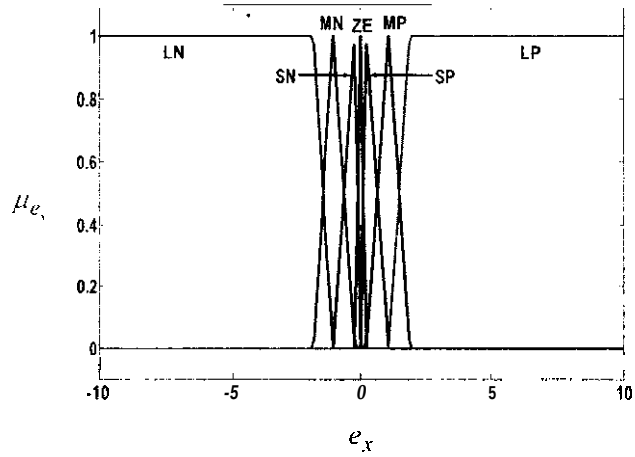


Figure 1 : Schematic of a FIS



Figurr 2: Membership functions for input e_x – example 1
(μ indicates degree of membership)

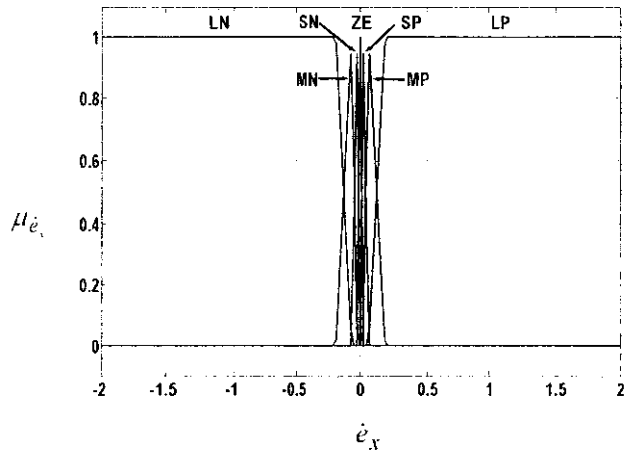


Figure 3: Membership functions for input $\dot{e}_x$ – example 1

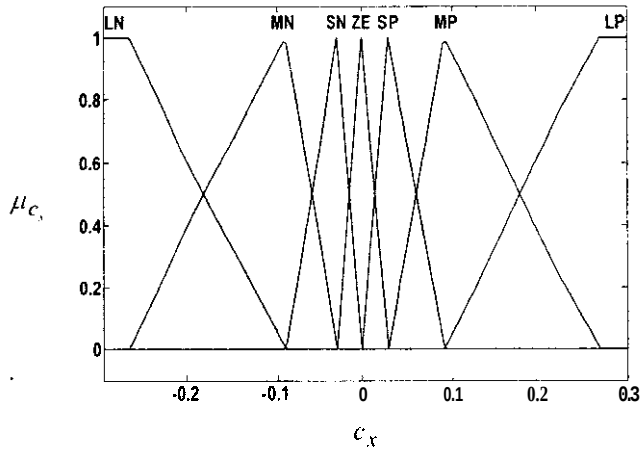


Figure **4** Membership functions for output c_x – example 1

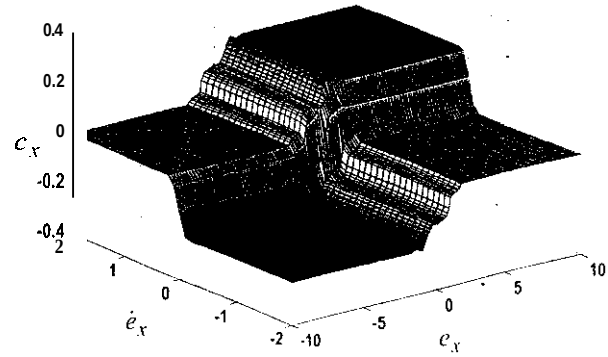


Figure 5: 3D surface view of inputs-output mapping – example 1

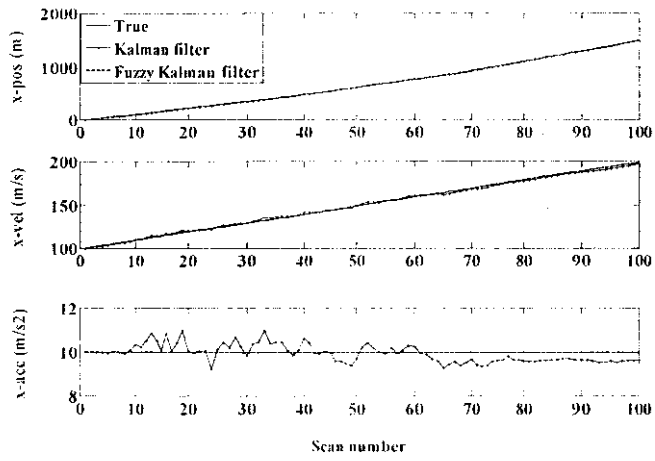


Figure 6 Comparison of true and estimated states – example 1

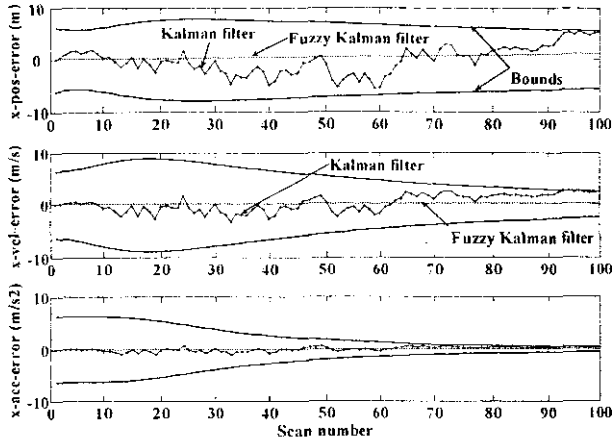


Figure 7: States errors with bounds – example 1

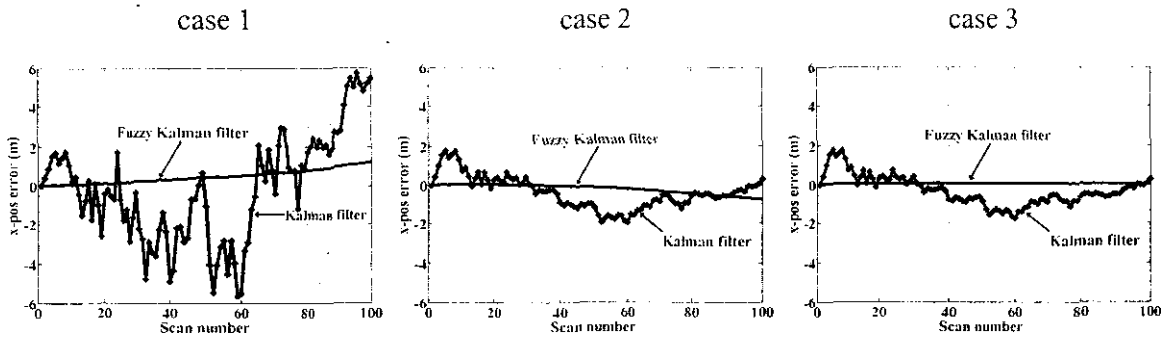


Figure 8: Comparison of position error for both the filters - example 1

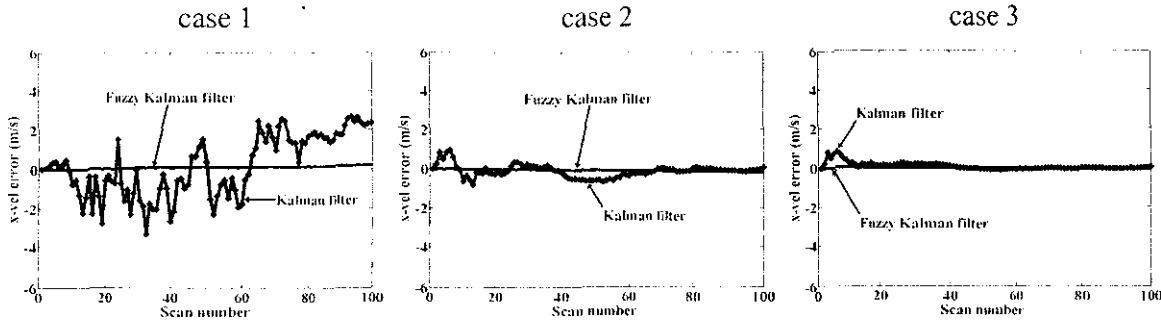


Figure 9: Comparison of velocity error for both the filters – example 1

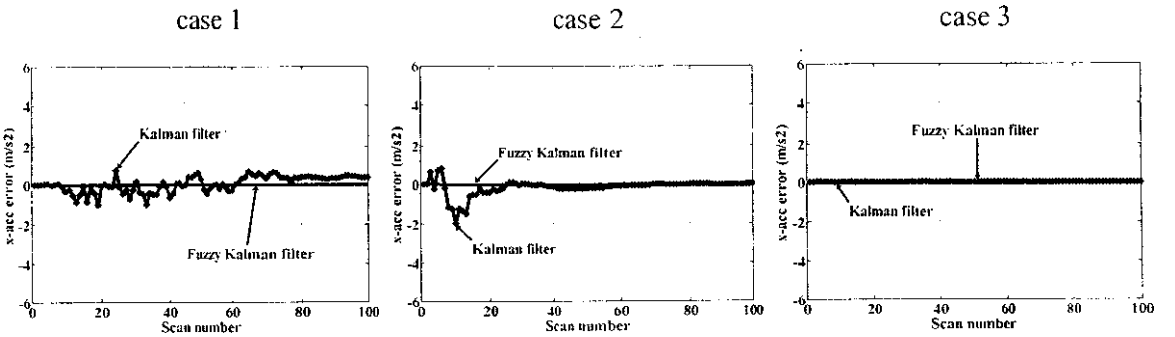


Figure 10: Comparison of acceleration error for both the **filters** – example 1

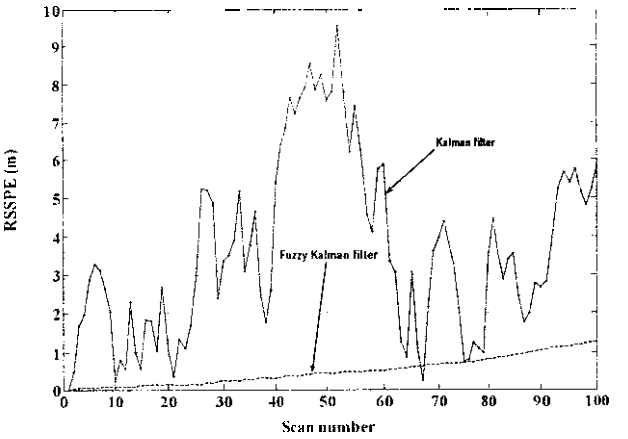


Figure 11: Comparison of RSSPE for both the filters – example 2

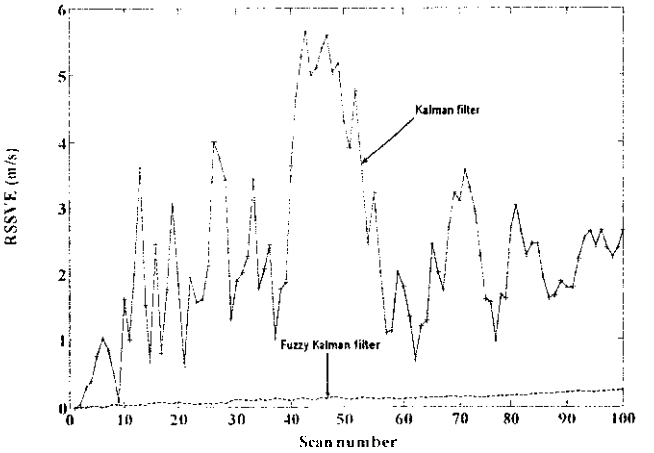


Figure 12: Comparison of RSSVE for both the filters – example 2

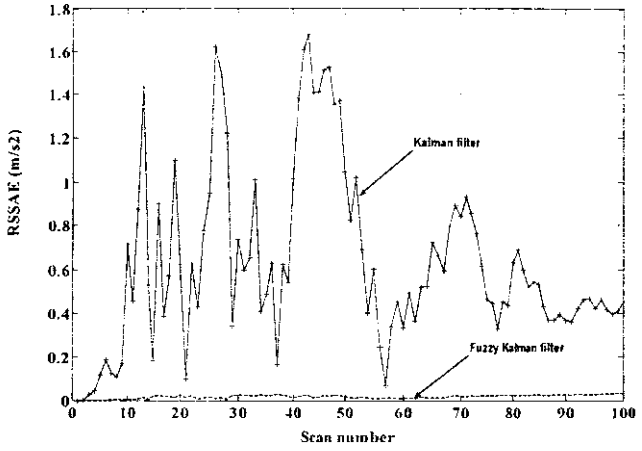


Figure 13: Comparison of RSSAE for both the filters – example 2

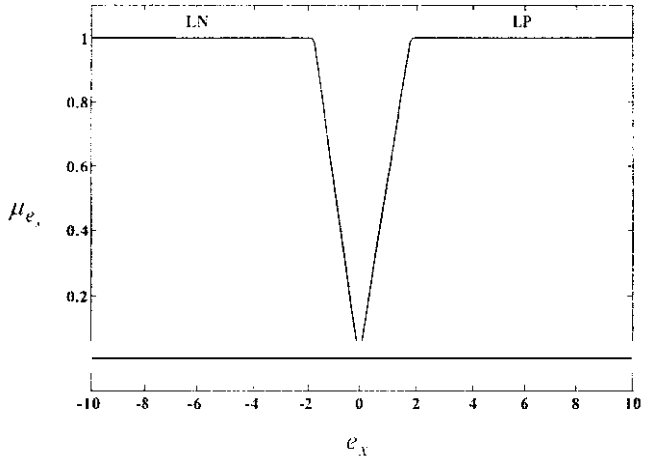


Figure 14: Membership functions for input e_x – example 3

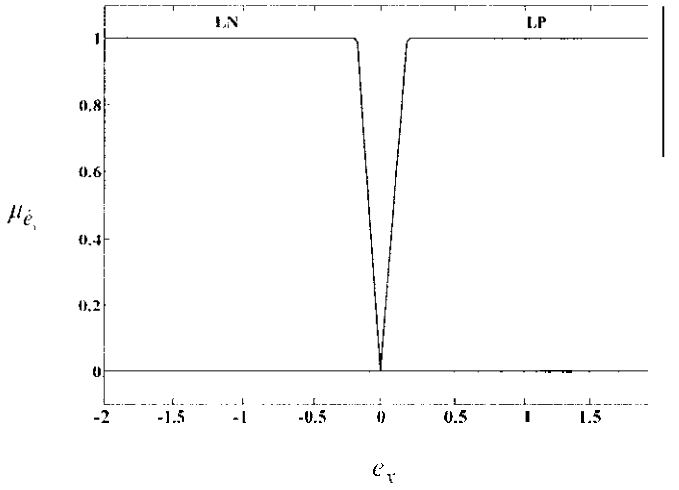


Figure 15: Membership functions for input $\dot{e}_x$ – example 3

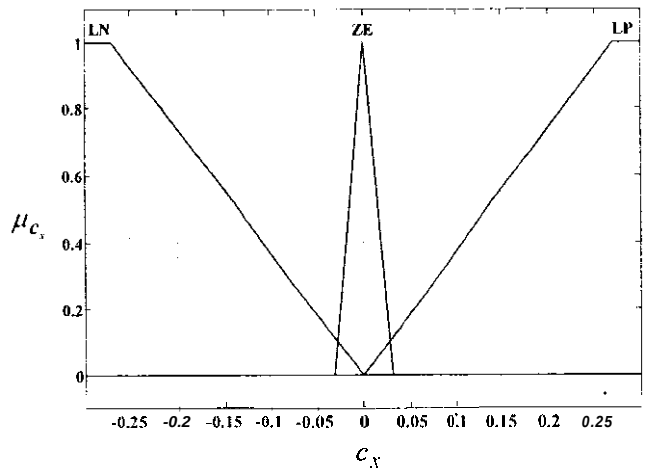


Figure 16: Membership functions for output c_X – example 3

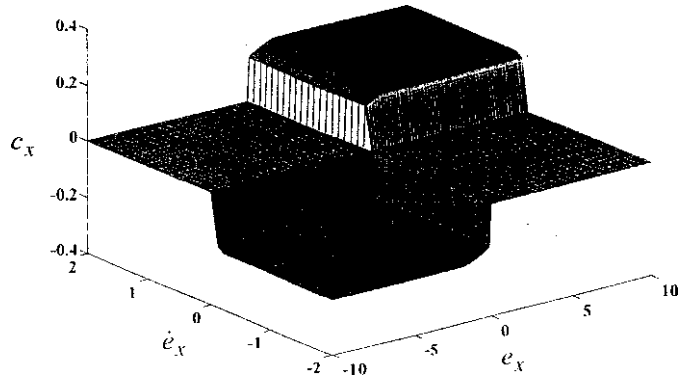


Figure 17: 3D surface view of inputs-output mapping – example 3

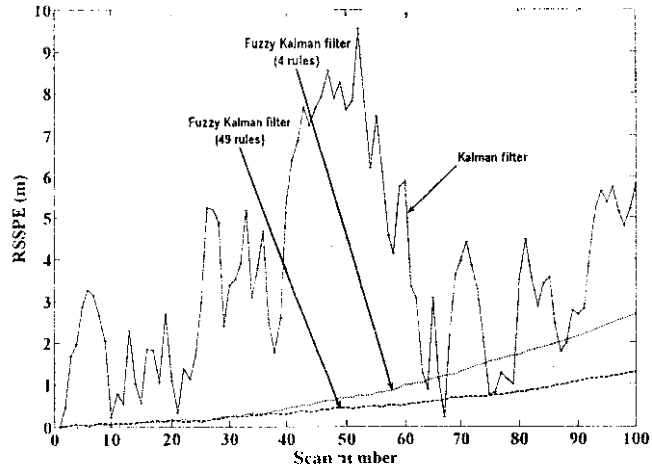


Figure 18: Comparison of RSSPE for both the filters – example 3

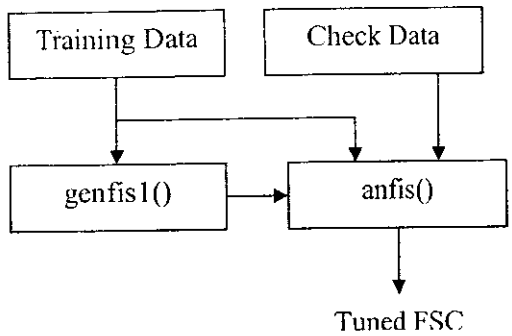


Figure 19 : Procedure to get tuned FSC – example 4

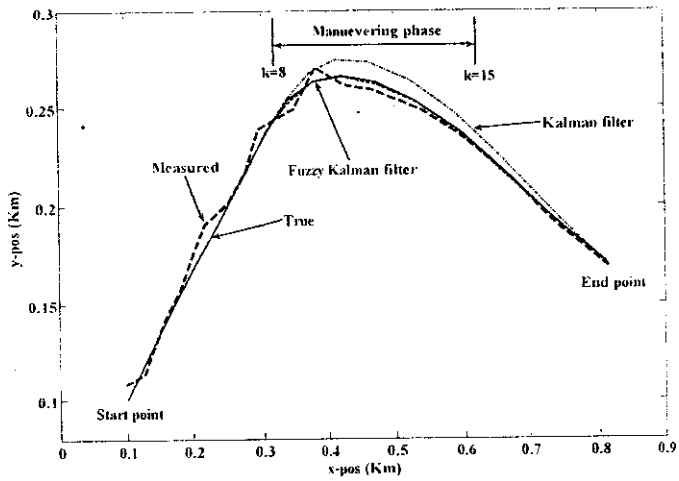


Figure 20 : Comparison of true, measured and estimated x-y target positions – Mild maneuver – example 4

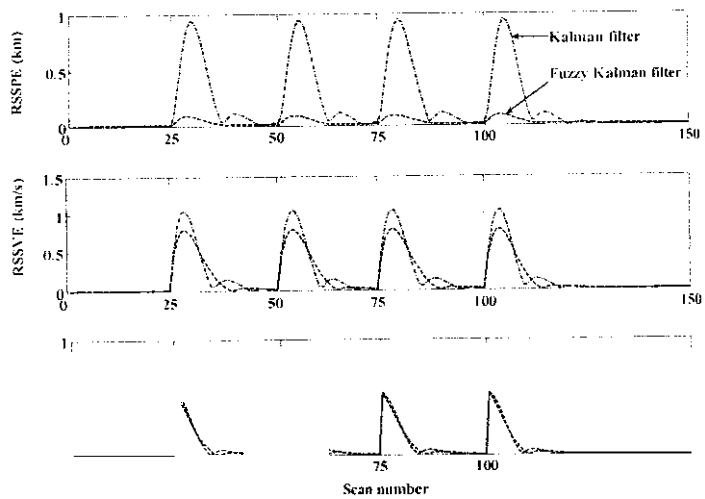


Figure 21: Comparison of RSSPE, RSSVE, and RSSAE for both the filters Evasive maneuver – example 4